Data Structures and Algorithms (DT505G)

Third Examination VT2020 (Exam #17)

Teacher:	Federico Pecora (ext. 3319)
Published:	2020-08-24 at 08:15
No. exercises:	6
Total points:	70 (42 required to pass)
No. pages:	10 (excluding this page)

- This is a **personalized exam**
- Compile your answers into a **Word** or **PDF** file named `17_submission.pdf`
- Submission instructions
  - Open the course Blackboard page, go to the Course material section
  - Click on the item entitled **Exam 2020-08-24 assignment**
  - Submit your Word/PDF file as an **attachment** to this assignment
- Submissions are due **at 12:15 today** (unless you are given another deadline by Funka)
- The teacher is available **from 10:00 to 11:00 to answer questions** at:
  - Zoom (browser): `https://oru-se.zoom.us/j/62806284525`
  - Tel.: `+46850500829` or `+46850520017` (meeting ID: 628 0628 4525)
  - Mob.: `+46850500829,,62806284525#` or `+46850520017,,62806284525#`
  - If you are put on hold, please wait, as this means there is a queue
- If further doubts arise, make reasonable assumptions and **write them down**

Exercise 1 (10 points)

Given the array $A = \langle 11, 16, 4, 13, 6, 10, 5, 8, 15, 7 \rangle$, show how the Insertion sort and Merge-sort algorithms work. Step through each algorithm, illustrating how it modifies the input array A. State the worst- and best-case computational complexity of the two algorithms in terms of the size $|A|$ of the input array, and explain why.

Exercise 2 (12 points)

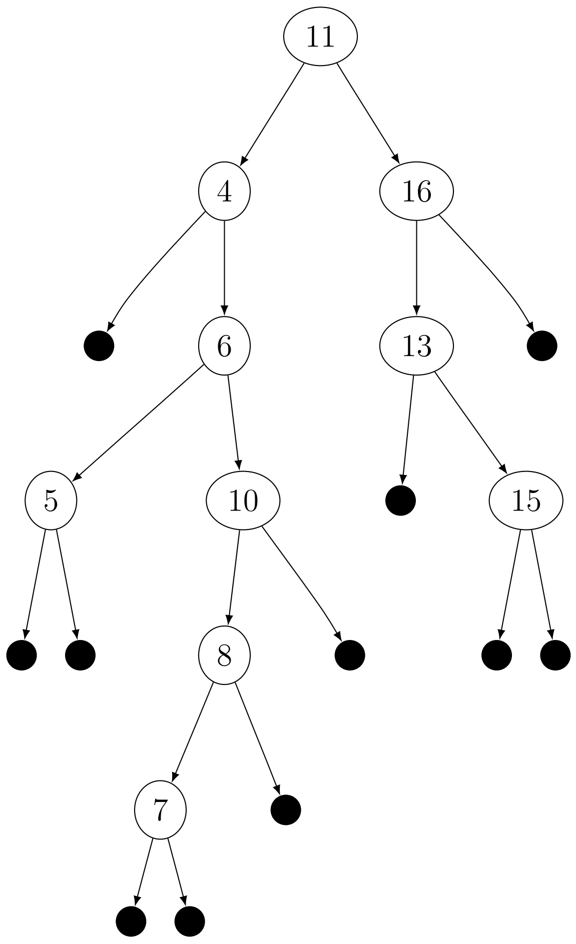
Construct a Binary Search Tree (BST) from the array of integers A in the previous exercise. Do this by inserting each integer into the tree, one by one, starting from the left (element '11'). Draw the resulting tree and state its height. Is the tree balanced?

Remove the third element of A (element '4') from the BST, explaining the operations that this procedure performs. Now, insert the element you removed back into the BST, and draw the resulting tree. Is the resulting BST different from the BST before the element was removed? If so, will this be the case for any element that is removed and then re-inserted?

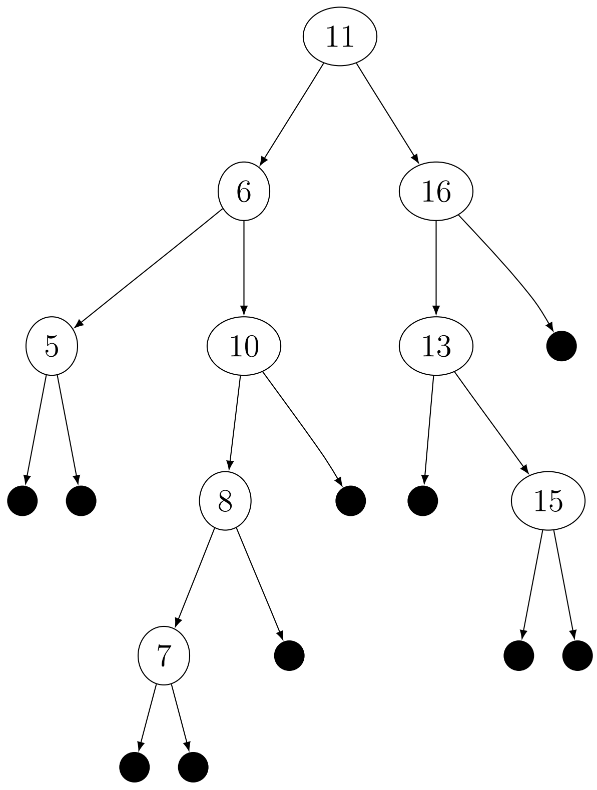
Illustrate the algorithm for finding the predecessor of an element in a BST. Show how this works, step by step, for the fourth element of A (element '13'). In general, how many key comparisons are performed by the algorithm for finding the predecessor of an element in a BST? What is the computational complexity of the algorithm?

Construct a new BST using the sorted array you obtained in Exercise 1, again by inserting the elements of the array into the tree one by one. What is the height of this BST, and is it balanced?

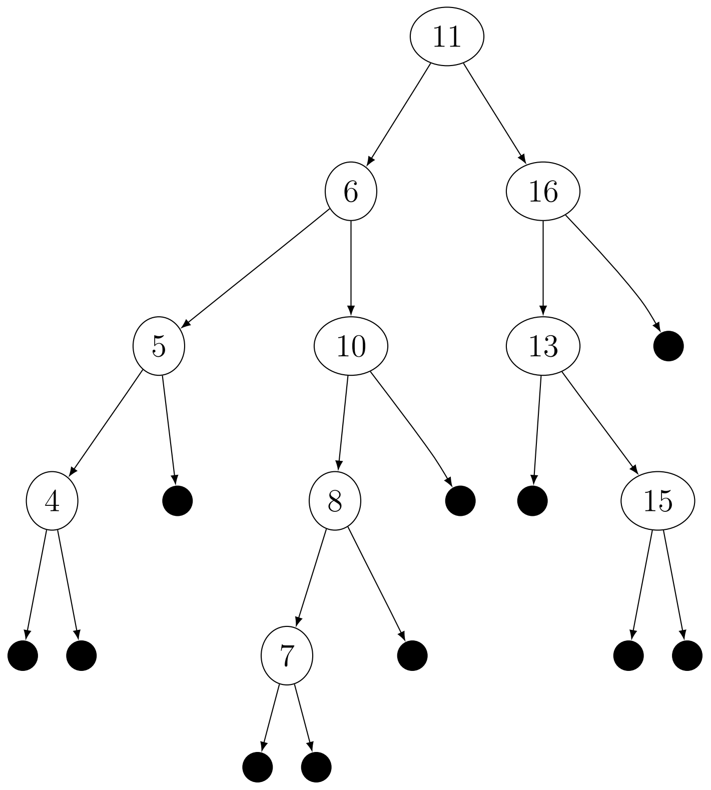
From A:



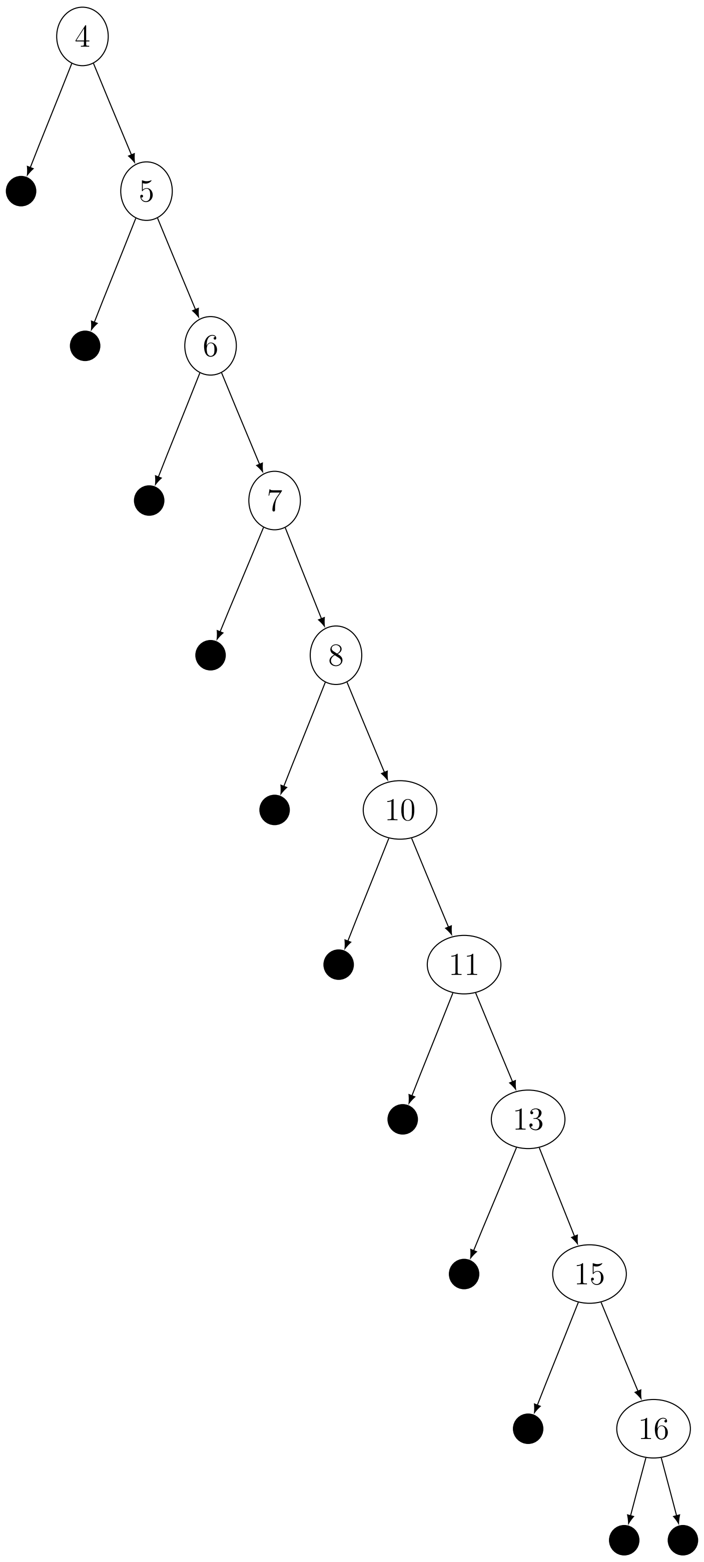
After removing ‘4’:



After re-adding ‘4’:



From sorted array:


4
5
6
7
8
10
11
13
15
16

Exercise 3 (12 points)

The *Collatz sequence* is defined as follows:

- start with any positive integer n;
- each term of the sequence is obtained from the previous term as follows:
  - if the previous term is even, the next term is one half of the previous term;
  - if the previous term is odd, the next term is 3 times the previous term plus 1.

For example, if we start with $n = 10$, we get the sequence $10, 5, 16, 8, 4, 2, 1, 4, 2, 1, \ldots$. Note that once the sequence reaches 1, it will forever cycle between the numbers $4, 2, 1$.

In 1937, the German mathematician Lothar Collatz stated the following conjecture:

Given any $n \in \mathbb{N}$, the Collatz sequence starting from n will always reach 1.

To this day, no mathematician has been able to prove or disprove this conjecture. Your task is to write a *recursive* algorithm to verify whether the Collatz sequence strating from a given number n reaches 1. The algorithm should take one argument (the number n to check) and print all the numbers in the sequence starting from n. The algorithm should terminate once the printed number is 1.

State the complexity of the algorithm as a function of the input number n in the case that n is a power of two, that is, $n = 2^k$ for some $k \in \mathbb{N}$.

```
COLLATZ($n$)
1  PRINT $n$
2  if $n == 1$ return 1
3  if $n \equiv 0 \pmod 2$ return COLLATZ($\frac{n}{2}$)
4  return COLLATZ($3n + 1$)
```

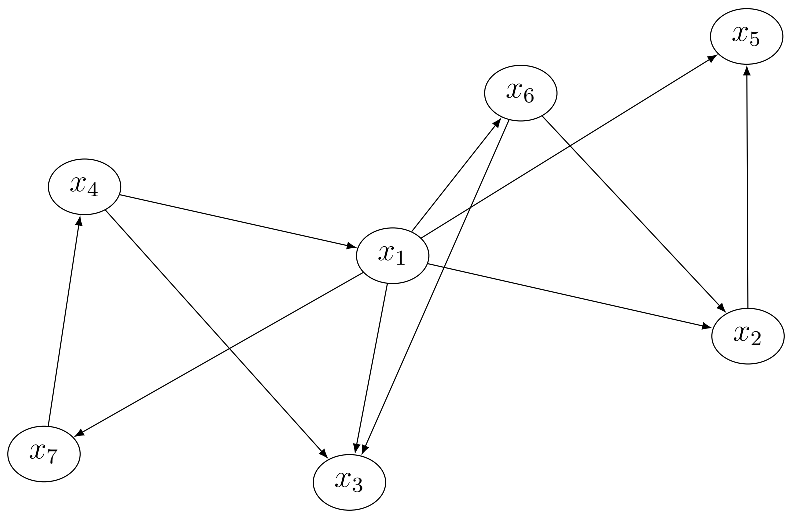
Complexity: $\Theta(\log n)$, because at each cycle, n will be divided by two.

Exercise 4 (12 points)

Given a directed, unweighted graph $G = (V, E)$, describe an algorithm to solve each of the following problems:

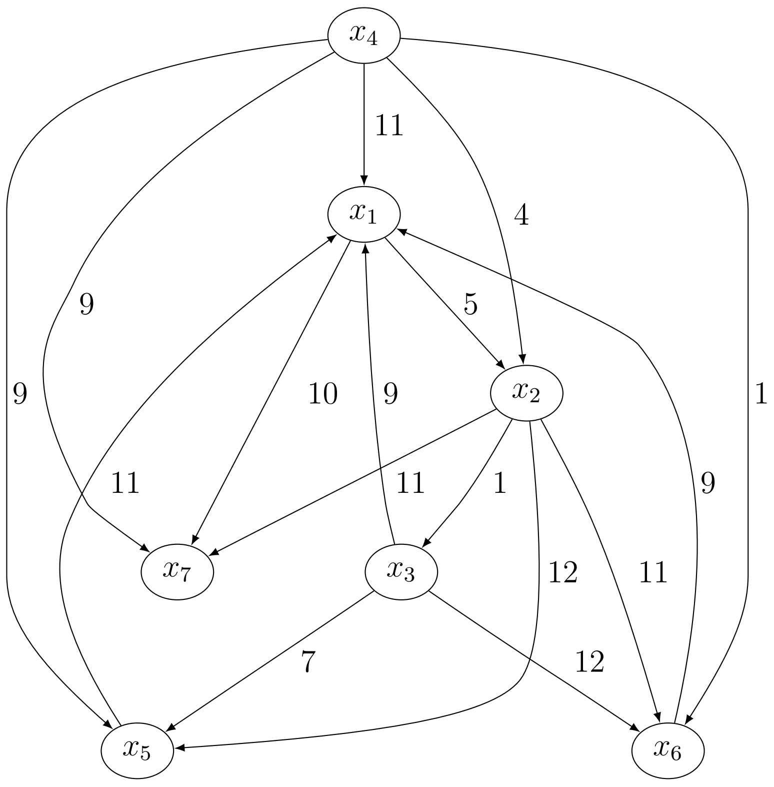
- Determine whether the graph is acyclic. DFS (iff no back edges), $\Theta(V + E)$
- Find all the strongly connected components of the graph. DFS + transpose + DFS, $\Theta(V + E)$
- Find the shortest path from node x_1 to every other node. BFS, $O(V + E)$

Show all the steps performed by each algorithm, using as input the graph G given below. Also, state and explain the computational complexity required to solve each problem in terms of the number of nodes $|V|$ and number of edges $|E|$ of the input graph.



Exercise 5 (12 points)

Given the following directed graph $G = (V, E)$ with positive weight funtion $w : E \mapsto \mathbb{N}$ as specified on the edges of the graph, illustrate an algorithm for finding a shortest path from vertex x_1 to every other vertex in the graph. Show how, at each step, the algorithm updates the distance estimate $d[x_i]$ of every vertex $x_i \in V$. State the computational complexity of the algorithm in terms of the number of nodes $|V|$ and number of edges $|E|$ of the input graph.



Distances: $[0, 5, 6, \infty, 13, 16, 10]$

Exercise 6 (12 points)

State whether each of the following 6 statements on the asymptotic behavior of the function f is true or false.

1.	$f(n) = 2n^4 + 2n^3 - 2n^2 - n$,	$g(n) = n\log(n)$,	$f(n) \in O(g(n))$	(false)
2.	$f(n) = 3^n$,	$g(n) = 2n^2$,	$f(n) \in \Omega(g(n))$	(true)
3.	$f(n) = n^3 - 3n^2 - 3$,	$g(n) = 3^n$,	$f(n) \in \omega(g(n))$	(false)
4.	$f(n) = 6^n$,	$g(n) = 2n^4 - 5n - 1$,	$f(n) \in \Omega(g(n))$	(true)
5.	$f(n) = 4n^2 - 4$,	$g(n) = 9n\log(n)$,	$f(n) \in \Omega(g(n))$	(true)
6.	$f(n) = n + 3$,	$g(n) = 2^n$,	$f(n) \in \Omega(g(n))$	(false)